Algorithm Analysis and Mapping Environment for Adaptive Computing Systems

Eric K. Pauer, Paul D. Fiore, John M. Smith, Cory S. Myers
Sanders, a Lockheed Martin Company
P.O. Box 868, Nashua, NH 03061-0868
Phone: 603-885-8358 Fax: 603-885-0631
{pauer, pfiore, jmsmith, cory}@sanders.com

Our team is developing an integrated algorithm analysis and mapping environment for migrating a dataflow representation of a signal processing algorithm into an Adaptive Computing System (ACS) consisting of field programmable gate arrays (FPGAs). This environment allows designers to transform signal processing algorithms into FPGA-based hardware faster, by an order of magnitude, than is currently possible. Our approach has been to focus on three areas of capability critical to the success of adaptive computing: algorithm analysis, algorithm mapping, and smart generators. These capabilities are taking advantage of the special characteristics of signal processing algorithms to reduce the time to field the ACS, and are being implemented as extensions to the Ptolemy design environment developed at the University of California, Berkeley.

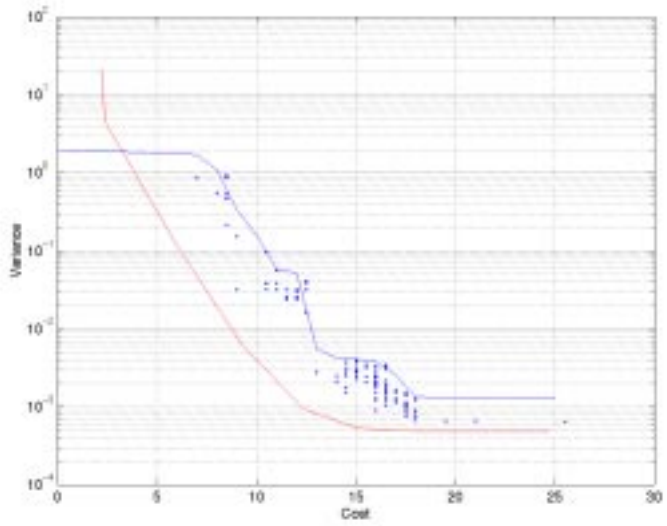

Figure 1: Quantization Noise versus Hardware Cost

Algorithm implementation for ACS requires careful consideration of the appropriate signal representation and the costs of operations. The algorithm analysis capabilities being developed on this program will reduce the effort required to find good ACS implementation choices for a signal processing algorithm. The environment will provide algorithm designers with information

about operation counts, including adds, multiplies, and memory accesses, and with analyses of quantization effects related to ACS implementations. For many DSP problems, reduced precision arithmetic will maintain acceptable system performance. A mapping of an algorithm to an FPGA architecture will be successful if the designer can limit wordlength growth without sacrificing algorithm performance. Wordlength reduction introduces noise into the data stream, so the designer must balance the need for an efficient implementation with output quality. Our environment supports both analytical and simulation-based wordlength optimization. With these capabilities an algorithm designer will be able to quickly determine the appropriate number of bits for signal representations at all points in the design and to quantify the performance of various implementation choices. Figure 1 shows an example of the relationship of quantization noise to hardware costs for a particular algorithmic dataflow graph.

Signal processing algorithm mapping for ACS involves the assignment of functions to different processing elements. On this program we are developing mapping techniques for ACS tailored to signal processing. These capabilities include performance analysis, partitioning assistance, and automatic scheduling and partitioning. The scheduling and partitioning functions will recognize the coarse-grain nature of signal processing dataflow graphs to optimize partitioning for ACS.

Current methods for logic generation for ACS are either built around libraries of functions or around general-purpose logic synthesis. As part of this effort, we are implementing "smart generators" that are extensions of the concept of a parameterized library. These generators will be tailored to signal processing functions and will include rules that capture specific implementation techniques and trade-offs. For example, a smart generator for a complex multiplier is able to trade between a three-multiplier implementation and a four-multiplier implementation according to area and latency constraints. Additionally, we are providing mechanisms to automatically generate both hardware and software interfaces for the resultant ACS. Our initial target system is a Xilinx XC4062XL board from Annapolis Micro Systems.

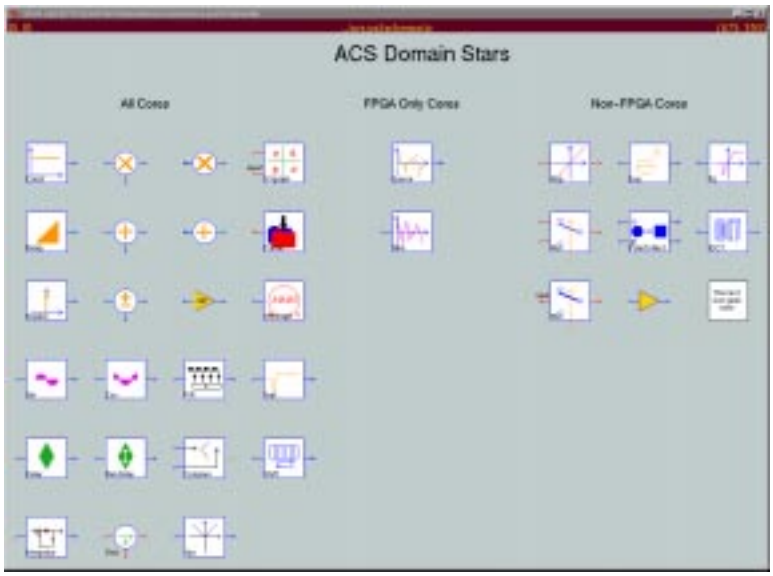

Figure 2: Adaptive Computing Functional Blocks (Stars)

The algorithm analysis, mapping, and logic generation capabilities are being developed as extensions to Ptolemy. Ptolemy provides a well documented, object-oriented, open software architecture with implementations in C++ and Java. Our extensions to Ptolemy are being captured in a new ACS domain that separates the interface specification from implementation for each signal processing functional block. The algorithms of interest to this project are represented by dataflow graphs comprised of these functional blocks (shown in Figure 2), following a synchronous dataflow model of computation. We are using a Corona/Core architecture, where each block has a common interface known as the Corona, and one or more implementations, known as the Cores. A retargeting mechanism allows the users change Cores and hence implementation, which moves the dataflow graph between various simulation models (floating point, fixed point) and implementations (C code, VHDL code).

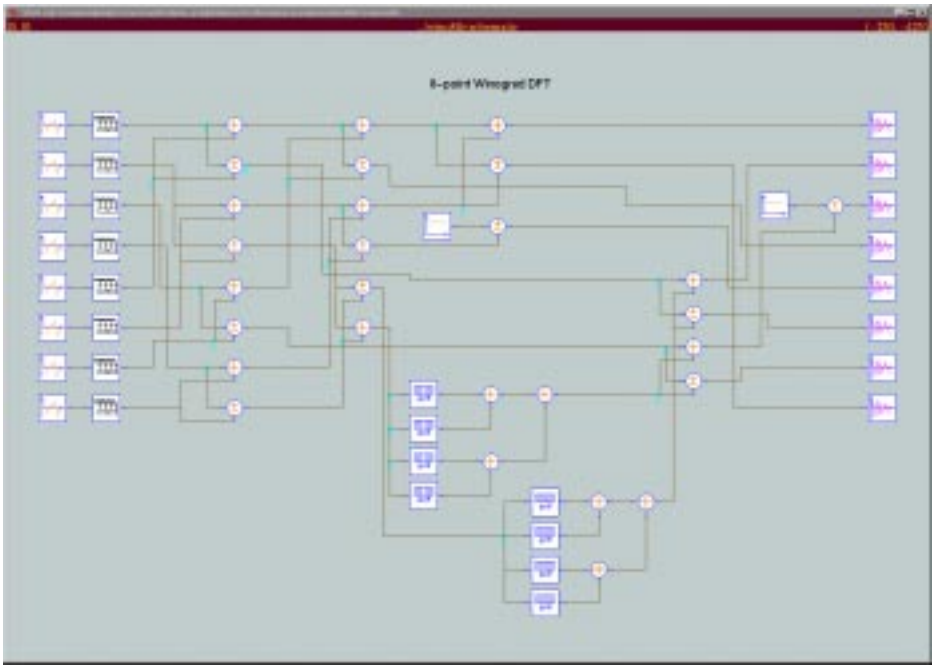

Figure 3: Winograd Discrete Fourier Transform (DFT) Dataflow Graph

Recently, these ACS tools have been used to automatically implement a Winograd DFT (see Figure 3) as part of a channelized FSK receiver in ACS. The Winograd algorithmic structure has the minimum number of multiplications for any DFT approach, and is thus ideal for FPGA implementation. The complete place and routed FPGA design is shown in Figure 4. The tools have also been used to develop an FPGA implementation of a high speed linear FM detector. In both cases, our ACS tools were used to simulate the algorithm, select appropriate fixed point representations, and generate the VHDL implementations. The final FPGA designs were obtained by synthesizing the VHDL and performing place and route with commercial tools. The ACS domain is part of Ptolemy 0.7.2 (released October 1999) and includes these ACS capabilities and demonstrations.

Future work includes extending the tools to handle multi-rate dataflow, multi-FPGAs, support for temporal and resource sharing in the optimization process, and support for multiple FPGA families. We are also planning on two more demonstrations: a military classification application and a high range resolution automatic target recognition problem.

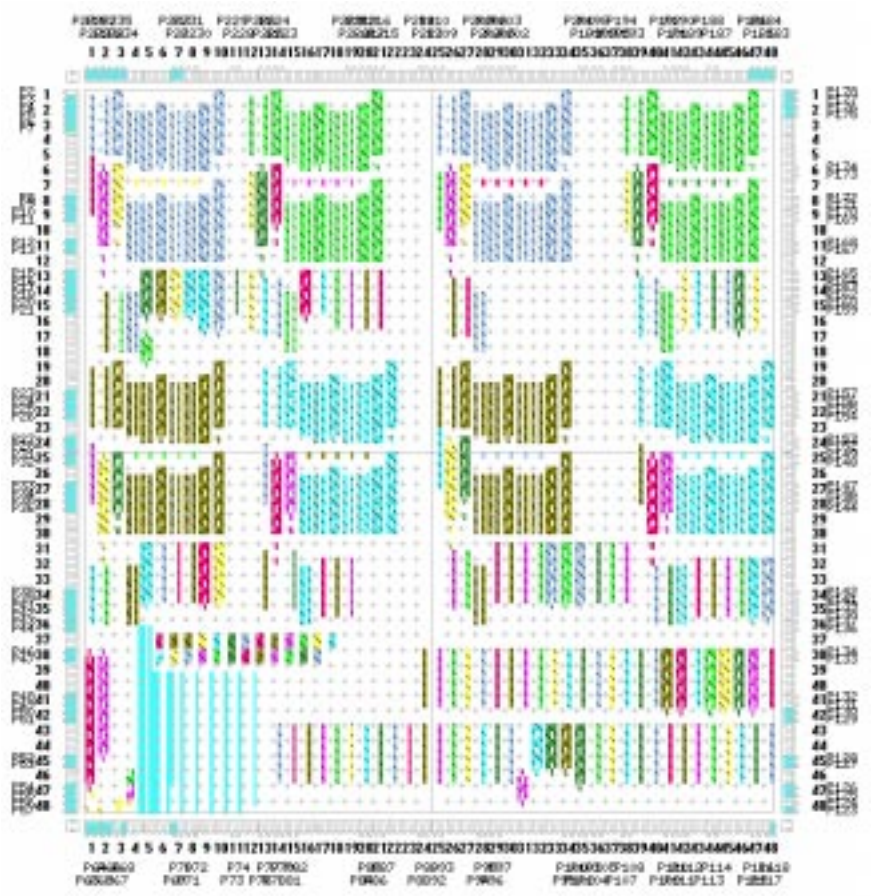

Figure 4: Completed Design for 8-point Winograd DFT in Xilinx 4062 FPGA

Acknowledgments: Portions of this work were supported by the Defense Advanced Projects Research Agency (DARPA) and the United States Air Force Research Laboratory (AFRL) under Contract No. F33615-97-C-1174.